KLS Vishwanathrao Deshpande Institute of Technology

(Accredited by NAAC with "A" Grade)

(Approved by AICTE, New Delhi, Affiliated to VTU, Belagavi)

(Recognized Under Section 2(f) by UGC, New Delhi)

Udyog Vidya Nagar, Haliyal - 581 329, Dist.: Uttara Kannada

www.klsvdit.edu.in | principal@klsvdit.edu.in | hodece@klsvdit.edu.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

University / Model Question Paper Scheme & Solution

Faculty Name	:	Rajeshwari Pashupatimath.
Course Name	:	Micro Controllers (BEC405A).
Course Code	:	BEC405A.
Year of Question Paper	:	2025 June July.
Date of Submission	:	20/1/2026.

Faculty Member

HoD

Head of the Department
Dept of Electronic & Communication Engg.

Dean (Acad.)

USN | | | | | | | | | | |

BEC405A

Fourth Semester B.E./B.Tech. Degree Examination, June/July 2025
Microcontrollers

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

			M	L	C
		Module – 1			
Q.1	a.	With diagrams, explain the RAM structure of 8051 microcontroller.	8	L2	CO1
	b.	With necessary sketches, explain: (i) Flags and program status word (ii) Stack operation.	8	L2	CO1
	c.	Write a note on Embedded Microcontrollers.	4	L1	CO1
		OR			
Q.2	a.	With a neat diagram, explain the block diagram of 8051 microcontroller.	8	L2	CO1
	b.	Write an interfacing diagram of 8051 microcontroller interfaced to 16K bytes of RAM.	8	L2	CO1
	c.	Compare CISC and RISC architecture.	4	L2	CO1
		Module – 2			
Q.3	a.	Write a program segment to copy the value 55 H into RAM memory locations 40 H to 44 H using, (i) Direct addressing mode, (ii) Register indirect addressing mode without a loop (iii) and with a loop.	6	L2	CO2
	b.	Explain the following instructions with examples: (i) Move A, @A + DPTR (ii) RRC A (iii) DA A.	6	L2	CO2
	c.	Briefly explain the arithmetics instructions of 8051 microcontroller.	8	L2	CO2
		OR			
Q.4	a.	Write an assembly language program to multiply the number present in external memory location 8000 AH and 8050 H. Store the lower byte of result obtained in R0 and higher byte in R1.	8	L3	CO2
	b.	Explain the role of CALL and subroutines in 8051 microcontroller programming. Give an example.	4	L2	CO2
	c.	If the number A6H is placed in external RAM between locations 0100H and 0200H, Write an assembly language program to find the address of that location and place that address in R6 and R7 registers.	8	L3	CO2
		Module – 3			
Q.5	a.	Explain the functions of each bit in the TMOD register.	6	L2	CO3
	b.	Explain MODE-1 programming of timers in 8051.	6	L2	CO3
	c.	Write a 8051 C program to transmit the message 'ECE' using serial communication port of 8051. Use baud rate 4800.	8	L2	CO3

Head of the Department
Dept. of Electronic & Communication Engg
KLS V.D.I.T. HALIYAL (U.K.)

		OR			
Q.6	**a.**	Explain the importace of TI flag and RI flag.	**6**	**L2**	**CO3**
	b.	Write the steps required for programming 8051 to transmit and receive the data serially.	**6**	**L2**	**CO3**
	c.	Explain how timers are used as counters and also explain the counters operation using a code snippet.	**8**	**L2**	**CO3**
		Module – 4			
Q.7	**a.**	Explain the following : (i) Interrupt (ii) Interrupt Service Routine (ISR) (iii) Interrupt Vector Table (IVT)	**8**	**L2**	**CO4**
	b.	Write the instructions to : (i) Enable the serial interrupt, timer 0 interrupt and external hardware interrupt. (ii) Disable the timer 0 interrupt. (iii) Disable all interrupts with a single instruction. Use bit manipulation instructions for all the cases.	**6**	**L2**	**CO4**
	c.	Explain the bit contents of IE register.	**6**	**L2**	**CO4**
		OR			
Q.8	**a.**	List the steps involved in executing interrupts in 8051 microcontroller.	**6**	**L2**	**CO4**
	b.	Assume XTAL = 11.0592 MHz. Use timer 0 to create the square wave. Write a assembly program that continuously gets a 8 bit of data from P(0) and sends it to P(1). While simultaneously creating square wave of 200 μs period on P2.5.	**8**	**L3**	**CO4**
	c.	Write the interrupt priority upon reset in 8051. Also explain how the priority of the interrupts can be set using IP register.	**6**	**L2**	**CO4**
		Module – 5			
Q.9	**a.**	With neat diagram, write an assembly language program to interface stepper motor to 8051 microcontroller.	**10**	**L3**	**CO5**
	b.	Explain DAC interface with diagram and also write program to generate triangular waveform.	**10**	**L3**	**CO5**
		OR			
Q.10	**a.**	With neat diagram, write an assembly language program to interface LCD to 8051 microcontroller.	**10**	**L3**	**CO5**
	b.	A door sensor is connected to the P1.1 pin and a buzzer is connected to P1.7. Write 8051 C program to monitor the door sensor and when it opens, sound the buzzer. The buzzer can be sound by sending a square wave of a few hundred Hz.	**10**	**L2**	**CO5**

* * * * *

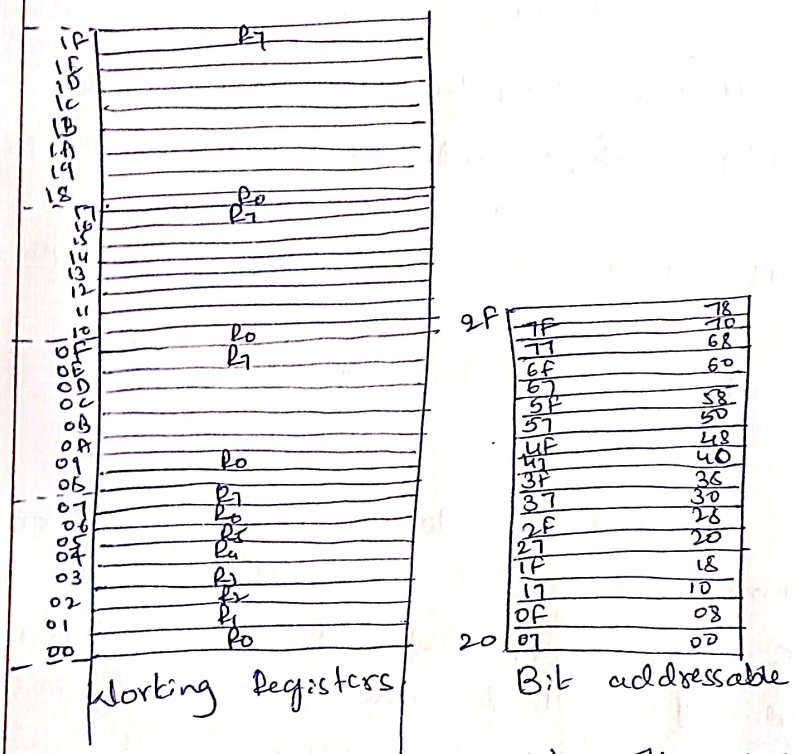
BEC405A.

Microcontrollers. , June/July 2025.

Module 1.

Q.1) (a) With diagrams, explain the RAM structure of 8051 microcontroller (8 marks)

Soln: The 8051 uC contains 128 bytes of internal RAM, whose address range from 00H to 7FH. This internal RAM is divided into 3 sections.



→ Register Banks (00H - 1FH). The area occupys 32 bytes and is divided into four register banks. Each bank contains 8 registers (R0-R7) The default bank selected after reset is Bank 0. Registers Bank selection is done using RS0 & RS1 bits of PSW.

→ Bit addressable RAM (20H-2FH): This section occupies 16 bytes. (128 bits) Each bit can be individually addressed and is mainly used for flag and control operations.

→ General Purpose RAM (30H-7FH): This section occupies 80 bytes. It is used for data storage, variable and stack operations. The stack pointer is initialized to 07H after reset.

Q(2) 02 b) With neat sketches, explain.

i) Flags and program status word. (8 masks).

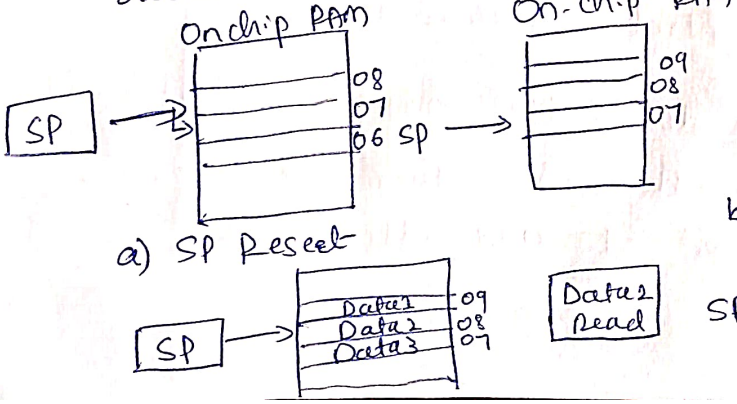
ii) Stack Operation.

Soln: i) Flags & PSW.

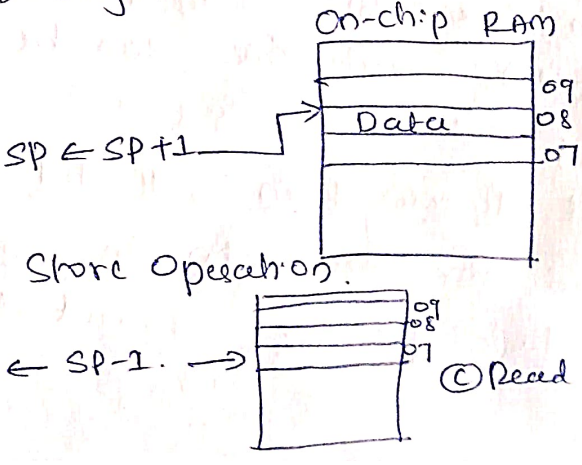
7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	–	P

- CY - Carry flag: It set when there is a carry out from MSB in arithmetic operation.
- AC (Auxiliary carry): Set when carry comes from lower nibble (D3 to D4), used in BCD operations.
- F0 (Flag): General purpose flag for use applications.
- RS1 & RS0 (Register Bank) select): used to select one of the 4 register banks.
- OV (Overflow flag): It set when signed arithmetic overflow occurs.
- P (Parity flag): Set if accumulator contains odd number of 1's

ii) Stack Operation: The stack refers to an area of internal RAM that is used to store and retrive data quickly. The Stack pointer (SP) register is used by 8051 to hold an internal RAM address that is called top of stack. The SP is 8 bit wide. It is incresed before data is stored during PUSH and CALL instructions and decremented after data is restored during POP & RET instns. The stack may be reside anywhere in on-chip RAM. The SP staste from 07H after reset. This cause the sp to begin at 08H.



a) SP Reset



c) Write a note on Embedded Microcontrollers (4marks) 03

Soln: The Embedded μC is a compact integrated circuit designed to perform a specific task with an embedded system. Unlike general purpose computers, embedded microcontrollers are dedicated to a single application and operate under predefined constraints such as limited memory, power, and processing speed.

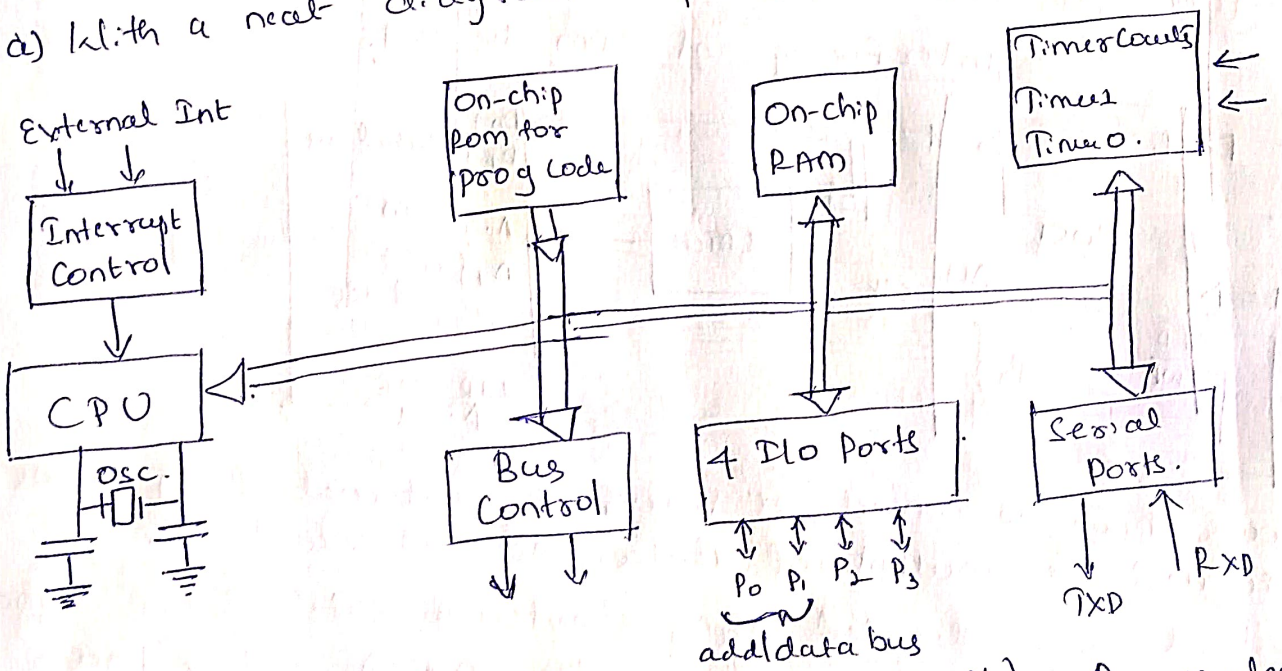
A typical μC consists of CPU, program memory (ROM/flash), data memory (RAM), I/P & O/P ports, timers, counters and communication interfaces (such as UART, SPI & I²C) all integrated on a single chip.

Embedded μC are widely used in consumer electronics (washing machines, microwave ovens), automotive systems (engine control unit, airbags), industrial automation, medical devices, and IoT applications.

OR

Q(2) a) With a neat diagram explain block diagram of MC. (8marks)

Soln:



1> CPU: Contains ALU (Arithmetic logic unit), Accumulator (A), B register, Program Status Word (PSW) and Prog counter (PC) for instruction execution and data manipulation.

2> Memory: ROM (Read only memory) Store temporary data, including general purpose register (R0-R7) and stack.

04 • RAM (Random access memory) : Stores temporary data (eg. 128 bytes internal), including general purpose registers (R0-R7) and stack.

3) I/O Ports: (P_0, P_1, P_2, P_3) : Four 8 bit ports for connecting external devices with some pins having dual functions

4) Timers/Counters (Timer 0 & 1) : Two 16 bit timers/counters for time delays or events counting.

5) Serial Port: for serial data communication (Tx & Rx).

6) Interrupt control: Manages various internal & external interrupt sources (eg INT0, INT1, Timers, Serial).

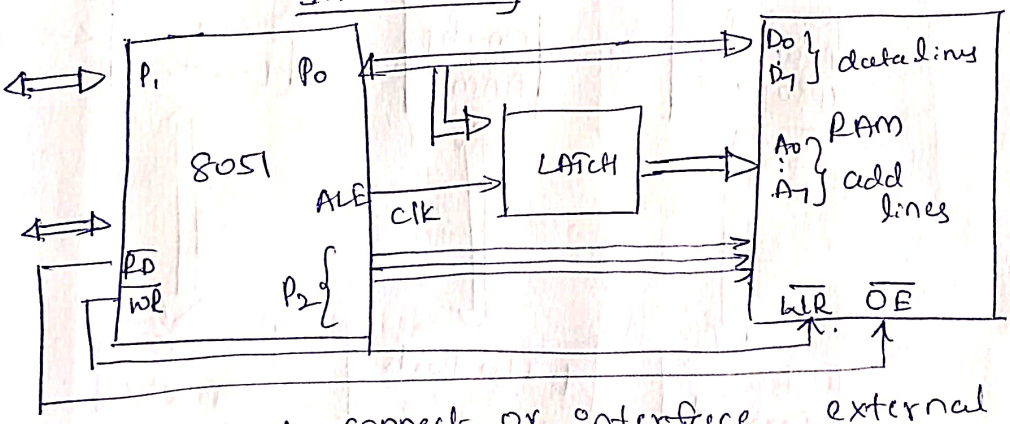
7) Oscillator & Clock circuit: Generates the system clock signal typically using external crystal pin XTAL1 & XTAL2.

8) Bus control: Manages data, address and control signals especially for external memory.

Q(2) b) Write an interfacing diagram of 8051 microcontrollers interface to 16k bytes of RAM (8 masks).

Interfacing of RAM to 8051.

Soln:



- Fig shows how to connect or interface external RAM to 8051
- Port 0 is used as multiplexed data and address lines.
- Address lines are decoded using external Latch & ALE signal from 8051 to provide lower order (A7-A0) address lines.
- Port 2 gives higher order address lines.
- RD & WR signals from 8051 selects the memory read and memory write operations resp.
- Generally P3.6 & P3.7 pins of port 3 are used to generate memory read & write signals. Remaining pins of Port 3 are used for other functions.

Q.(2) Q. Compare CISC & RISC architecture. (4marks) 05

Soln:

	CISC	RISC
①	It has simple instructions set.	It has complex instruc - ons set.
②	Execution Time is less, most instns executed in a single cycle.	Instructions can take multiple clock cycles.
③	Highly dependent on pipelining for performance. Its easier to implement.	less dependent on pipelining it is more complex to implement effectively.
④	Large no of general purpose registers	Small no of general purpose registers.

Module-2

Q(3) a) Write a program segment to copy the value 55H into RAM memory location 40H to 44H using
i) Direct add mode ii) Register indirect add mode.
iii) with a loop. (6marks).

Soln: 1> Direct addressing mode.

```
MOV. 40h, #55h
mov 41h, #55h
MOV 42h, #55h
MOV 43h, #55h
mov 44h, #55h.
```

2> Reg Indirect Mode.

```
mov R0, #40h
mov A, #55h
mov @R0, A
INC R0
mov@R0, A.
INC R0
mov @R0, A
INC R0
mov @R0, A.
INC R0
mov @R0, A.
```

3> Using loop.

```
MOV R0, #40h
MOV A, #55h
MOV R1, #05h
LOOP_LABEL
MOV @R0, A.
INC R0
DJNZ R1, LOOP_LABEL.
```

Q(3) b) OR Explain the following instructions with examples. (6marks)

i) Move A, @A+DPTR

ii) RRC A.

iii) DAA.

Soln: ① MOV A, @A+DPTR.

Type: Indirect add with displacement

Purpose: It moves data from memory locations whose address is calculated as A + DPTR into the accumulator A.

```
MOV DPTR, #3000h ; DPTR points to 3000h
mov A, #05h      ; offset 5
MOV A, @A+DPTR ;  A = [3000h+5] = [3005h]
```

② RRC A (Rotate Right through Carry).

Type: Bit manipulation instruction.

Purpose: Rotates all bits of accumulator (A) right by one bit, passing the LSB into Carry flag & taking previous carry into the MSB.

Ex: Before A = 10110010, CY=1

RRC A executed.

After: A = 11011001, CY=0.

③ DAA: (Decimal Adjust Accumulator)

Type: Arithmetic instruction (used before BCD addition)

Purpose: Adjust the 'A' so that the result becomes valid BCD

```
MOV A, #27h ; BCD = 27
ADD A, #35h. ; 27+35 = 5C (Hex) → invalid BCD.
DAA.         ; Adjust → A = 62h (BCD 62).
```

Q(3)(c) Briefly arithmetic instructions of 8051 uC. (8marks)

Soln: 1> ADD (addition): Syntax: ADD A, # data.

Operation: Adds source to accumulator and store result in A.

Flags effected: CY, AC & OV.

Ex:

```
MOV A, # 25h.
ADD A, # 12h ; A = 37h.
```

2) ADDC (Add with carry).

Syntax: ADDC A, #data,

Opration: Add source +cy to accumulator

Ex: MOV A, #25h

SETB cy ; cy=0.

ADDC A, #12h ; A = 25h +12h +1 = 38h.

3) SUBB (Subtract with Borrow)

Syntax: SUBB A, #data

Operation: subtract source + cy from accumulator.

Ex: MOV A, #50h

CLR cy ; Carry =0.

SUBB A, #20h; A = 50h - 20h = 30h.

4) INC / DEC: (Increment / Decrement)

Syntax: INC A / DEC A or INC Rn / DEC Rn.

Operation: Add 1 (INC) or subtract 1 (DEC) from Operand.

Ex: MOV A, #0Ah

INC A ; A=0Bh

DEC A ; A = 0Ah.

5) MUL AB. (Multiply)

Syntax: MUL AB

Operation: Multiplies A & B result stored as High byte: B, Lowbyte: A.

Flags: cy & OV set if result > 255

MOV A, #05h

MOV B, #04h

MUL AB ; A = 14h, B = 00h

6) DIV AB (Divide)

Syntax DIV AB

Operation : Divide A by B.

Quotient → A, Reminder → B , Flags: cy cleared.

Ex: MOV A, #14h

MOV B, #05h

DIV AB ; A = 02h (Q) , B = 04h (R).

Q4(a) 08 Write an assembly language program to multiply the numbers present in external memory location 800 AH and 8050 H. Store the lower byte of result obtained in R0 and higher byte in R1. (8marks)

Soln:

```
Org 0000h
Mov DPTR, #800Ah
MOVX A, @DPTR
Mov B, A
Mov DPTR, #8050h
MOVX A, @DPTR
MUL AB
MOV R0, A
MOV R1, A
SJMP HALT
END.
```

b) Explain the role of CALL and subroutines in 8051 microcontroller programming. Give an example. (4marks).

Soln: A subroutine in 8051 uc is a small block of program written to perform a specific task. It can be used many times in a program, thus reducing code size.

A CALL instruction is used to transfer control from the main program to the subroutine. In 8051, 2 CALL instrns are available,

- ACALL - Calls a subroutine within the same 2kB memory block
- LCALL - Calls a subroutine anywhere in 64kB pgm memory.

When a CALL instn is executed, the address of the next instn is stored in the stack and control jumps to the subroutine After execution the RET instn returns control to the main program.

Ex:

```
SETB P1.0
ACALL DELAY   Here, DELAY is a subroutine
CLR P1.0      called using CALL instn.
```

Q4(c) If the number A6H is placed in external RAM between locations 0100h and 0200h write an assembly language program to find the address of that location and place that address in R6 & R7 registers. (8marks). 09

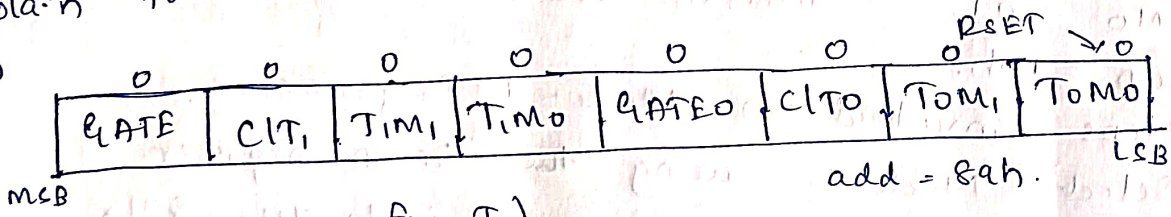
Soln:

```
        Org  0000h
        Mov  DPTR, #0100h
SEARCH: MOVX A, @DPTR
        CJNE A, #0A6h, NEXT
        Mov  R6, DPL
        MOV  R7, DPH
        SJMP DONE
NEXT:   INC  DPTR
        MOV  A, DPH
        CJNE A, #02h, SEARCH
DONE:   SJMP DONE
        END.
```

MODULE-3

Q(5) a) Explain the functions of each bit in TMOD register (8marks).

Soln: TMOD



① GATE (Bit 3 for T0, Bit 7 for T1)
•1: Timer/Counter runs only when TRx (Timer Run) bit in TCON is set and the external pin (INT0 for T0 & INT1 for T1) is high.
•0: Timer/Counter runs when ever TRx is set, regardless of the external pin (s/w control).

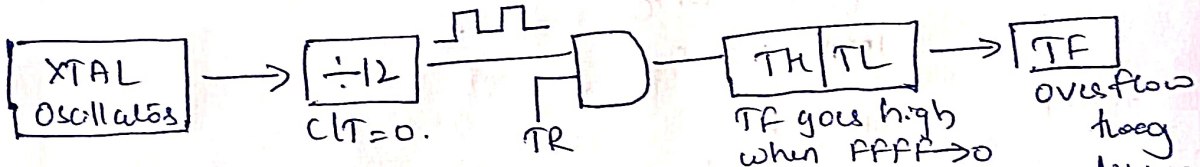
② C/T (Bit 2 for T0, Bit 6 for T1)
•1: Counter Mode (Counts external events on Tx pin)
•0: Timer mode (Counts internal machine cycles).

③ M1 & M0. (Bits 1, 0 for T0; Bits 5, 4 for T1): These two bits select one of four operating mode.

M1	M0	Mode	function.
0	0	Mode0	13 bit Timer (Counter. CTHx + 5bit TLx)
0	1	Mode1	16 bit Timer (Counter CTHx & TLx)
1	0	Mode2	8 bit Auto Reload.
1	1	Mode3	Split Timer mode.

Q5(b) Explain MODE-1. programming of Timers in 8051. (6 marks)
(6 marks)

Soln:



In the 8051 microcontroller, timer can operate in different modes. Mode-1 is a 16bit timer/counter mode in which the timer uses two 8 bit registers, THx & TLx.

In this mode, the timer counts from the loaded initial value up to FFFFh. When the count rolls over from FFFFh to 0000h, the overflow flag TFx is set.

M1 M0 = 01. Timer 0 Mode 1 → TMOD = 01h, Timer 1. Mode 1 → TMOD = 10h.

Steps involved in Mode-1 Programming.

1) Select Mode-1. using the TMOD register.
2) Load the initial count value in to THx & TLx
3) Start the timer by setting the TRx bit in TCON reg.
4) The timer increments with each machine cycle.
5) When Timer overflows, TFx flag is set.
6) Clear TFx after overflow.

Q5(c) Write a 8051 C program to transmit the message 'ECE' using serial communication port of 8051. Use baud rate 4800 (8marks). 11

Soln:

```
# include <reg 51.h>
Void UART_Init();
    TMOD = 0X20;
    TH1 = 0XFA;
    SCON = 0X50;
    TR1 = 1;
}
void Transmit_data (char tx_data)
{
    SBUF = tx_data;
    while (TI == 0);
    TI = 0;
}
void Transmit_string (char *str)
{ int i;
  for (i=0; str[i] != 0 ; i++)
  { Transmit_data (str[i]);
  }
}
    void main {
      UART_Init();
      Transmit_string ("ECE");
      while (1) {
    }
}
```

Q(6) a) Explain the importance of TI flag & RI flag (6 marks)

Soln: TI (Transmit Interrupt flag)

- TI is set to 1 by hardware when one byte of data has been completely transmitted through the serial port.
- It indicates that Transmit buffer (SBUF) is empty and ready for the next data.

- It must be cleared by software after transmission.
- It helps in monitoring & controlling serial data transmission.
- TI ensures reliable data transmission by informing the program that the previous data byte has been sent successfully.

RI (Received interrupt) flag:

- RI is set to 1 by hardware when one byte of data is completely received through the serial port.
- The received data is stored in SBUF register.
- RI must be cleared by software after reading the data.
- It allows the program to detect incoming data immediately.
- RI also generate serial interrupt.
- RI helps in proper reception of serial data by indicating the valid data is available in SBUF.

Q6(b) Write the steps required for programming in 8051 to transmit and receive the data serially. (6marks).

Soln: Steps for Serial Communication.

1) Transmission:
- Configure the TMOD register to select Timer 1 in Mode-2 for baud rate generation.
- Load the TH1 register with the baud rate value.
- Configure the SCON register to select serial mode and enable transmission.
- Start Timer 1 by setting TR1 bit.
- Load the data byte into SBUF register to begin transmission.
- Wait for the TI flag to become 1.
- Clear the TI flag after sending the data.

2) Reception:
- Configure the SCON register to enable reception (REN=1)
- Wait for RI=1.
- Read the received data from SBUF reg.
- Clear the RI flag.

TMOD - Timer mode selection, TH1 - Baud rate setting.
SCON - Serial control SBUF - Serial data buffer.
TI & RI - Transmit and Receive flags.

Q6(b) ~~Write the steps~~ Explain how timers are used as counters and also explain the counters operation using a code snippet. (5marks).

Soln: Timers become counters by switching their i/p source from the internal system clock to an external pin. Incre-menting the internal register with each pulse on that pin instead of a machine cycle, allowing them to count external events like button presses or pulses from other sensors. config-ured via register like TMOD & TCON in micro controllers like the 8051, using C code to set the CT bit & starting the counters.

When Timer act as Counter.

1) A std timer counts internal clock cycles. To make it a counter, a configuration bit CT in the 8051 is set, telling the hlw to use an example pin as clock source.
2) Each detected transition increments the internal timer register by one.
3) Each detected transition increments the enternal timer reg.
4) Counters typically count upwards until they overflow. (eg from FFFFh → 0000h in 16 bit mode) & set a flag. (TFx).

Counter pgm example using a code snippet:

```
#include <reg51.h>
void main() {
    TMOD = 0x05;
    TL0 = 0x00;
    TH0 = 0x00;
    TR0 = 1;

    while(1) {
    if (TF == 1) {
        COUNT_LED = TL0;
        TF0 = 0;
    }
}
}
```

MODULE - 1.

Q(7) a) 14 Explain the following
i) Interrupt
ii) Interrupt Service Routine. (ISR)
iii) Interrupt vector Table. (remarks).

Soln: 1) Interrupt: An interrupt is a signal that temporarily stops the normal execution of a mc pgm & transfers to a special routine to handle an urgent task.

When an interrupt occurs: The CPU suspends the main pgm and saves the current pgm status. Next executes ISR. Then returns back to the main pgm after completion.

Types of Interrupts.

1) Hardware Interrupts: Generated by external devices
2) Software Interrupts: Generated by instructions in the pgm.
3) Maskable Interrupts: Can be enabled or disabled by CPU.
4) Non-maskable Interrupts: Can't be disabled.

2) ISR: Its a special function or sub pgm that is executed automatically when an interrupt occurs. The ISR stored at a specific memory location & execute based on Interrupt priority. Ends with a return from interrupt instn (RETI).

ISR Execution: Interrupt occurs, CPU saves the PC & status. ISR executed. RETI restores the main pgm.

3) IVT: Its a table stored in memory that contains the starting add of all ISRs. Each Interrupt has fixed vector add. When an interrupt occurs, the CPU identifies the interrupt, Fetches ISR add from IVT & jumps to the add.

Includes (IVT. 8051)

Q(7) b) Write the instructions to
i) Enable the serial interrupt, times 0 interrupt and external hardware interrupt.
ii) Disable the Times0 interrupt
iii) Disable all interrupts with a single instn. (6marks).
Use bit manipulation instn for all the cases.

Soln: 1) Enable a) Serial Int b) Times 0 Int c) INT0.

Bit	Symbol	function
IE.7	EA	Global IE
IE.4	ES	Serial IE
IE.1	ET0	Times 0 IE
IE.0	EX0.	External Int 0 Enable.

In 8051 SETB instn using to set all the instns.

2) Disable Times 0 Interrupt.
Times 0 interrupt enable bit = ET0 (IE.1)
Instn: CLR IE.1 ; Disables Times 0 interrupt.

3) Disable all interrupts with a single instn.
All interrupts are controlled by the EA (Enable all) bit.
Instn: CLR IE.7 ; Disable all interrupts.

Q(7) c). Explain the bit contents of IE register. (6marks)

Soln:

EA	-	-	ES	ET1	EX1	ET0	EX0
D7	D6	D5	D4	D3	D2	D1	D0

D7 EA → Enable all interrupts EA=1, Disable all interrupts EA=
D6 & D5 → Reserved not used.
D4 → ES → Enable / Disable Serial Interrupt.
D3 → ET1 → Enable / disables Times 1 Int.
D2 → EX1 → " " External Int 1 (INT1)
D1 → ET0 → " " Times 0 int.
D0 → EX0 → " " External Int 0 (INT0).

Q(4) a) 16 List the steps involved in enabling interrupts in 8051 µC. (8 marks)

Soln: 1> An interrupt request is generated either by an external hardware signal (INT0 / INT1) or by an internal event such as timer overflow or serial comm^n..

2> The 8051 completes the execution of the current instn before responding to a interrupt.

3> The CPU checks whether
- The global int enable bit EA is set.
- The corresponding interrupt enable bit in the IE reg is set. & No higher priority int is being serviced.

4> The contents of the PC are automatically pushed onto the stack to save the return add.

5> Branch to Int vector add: The CPU loads the PC with the Int vector add corresponding to the int & starts executing the ISR.

6> The ISR performs the required task related to int.

7> At the end using RET instn pops the saved PC from the stack. & resume to the main pgm.

Q 4(b) Assume XTAL = 11.0592 MHz. Use Timer 0 to create the square wave. Write a assembly pgm that continously gets a 8 bit of data from P1.0 and sends it to P1.1. While simultaneously creating square wave of 200µs period on P2.5. (8 marks),

Soln: Machine cycle Time: 11.0592 MHz / 12 = 921.6 kHz

⇒ 1 / 921.6 kHz ≈ 1.085 µs. / machine cycle.

100 µs / 1.085 µs/tick ≈ 92 ticks.

Reload value: Timer 0 (16 bit) overflows after 65536 counts. For 92 ticks = 65536 - 92 = 65444 = FF84.

TH0 = 0xFF & TL0 = 0x84

```
        org 0000h
        MOV TMOD, #01h
        SETB ET0
        SETB EA
        SETB TR0
        MOV P1, #00h
MAIN-LOOP:
        MOV A, P0;
        MOV P1, A.
        SJMP MAIN-LOOP
        org 000Bh
TIMER0-ISR:
        CPL P2.5
        MOV TH0, #0FFh
        MOV TL0, #084h
        RETI
        END.
```

Q8(c) Illustrate the interrupt priority upon reset in 8051. Also explain how the priority of the interrupts can be set using IP registers. (6marks).

Soln: When 8051 uC is reset the interrupt Priority (IP) register is cleared to 00h. This means all the interrupts have the same priority level (low priority). In this condition, if two or more interrupts occur simultaneously the fixed hardware polling sequence determines the order of serving.

Default Priority Order (After Reset).

1) External Int 0 (INT0)
2) Timer 0 Interrupt
3) External Int 1 (INT1)

18 4> Timer 1 Int
5> Serial Int.
The order is also the interrupt polling sequence of 8051.

IP register.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
-	-	-	PS	PT_1	PX_1	PT0	PX0

Bit	Symbol	Function.
D_7, D_6, D_5	-	Reserved.
D_4	PS	Serial Int Priority
D_3	PT_1	Timer 1 Int Priority
D_2	PX_1	Ext Int 1 Priority
D_1	PT0	Timer 0 int priority
D_0	PX0	Ext Int 0 Priority

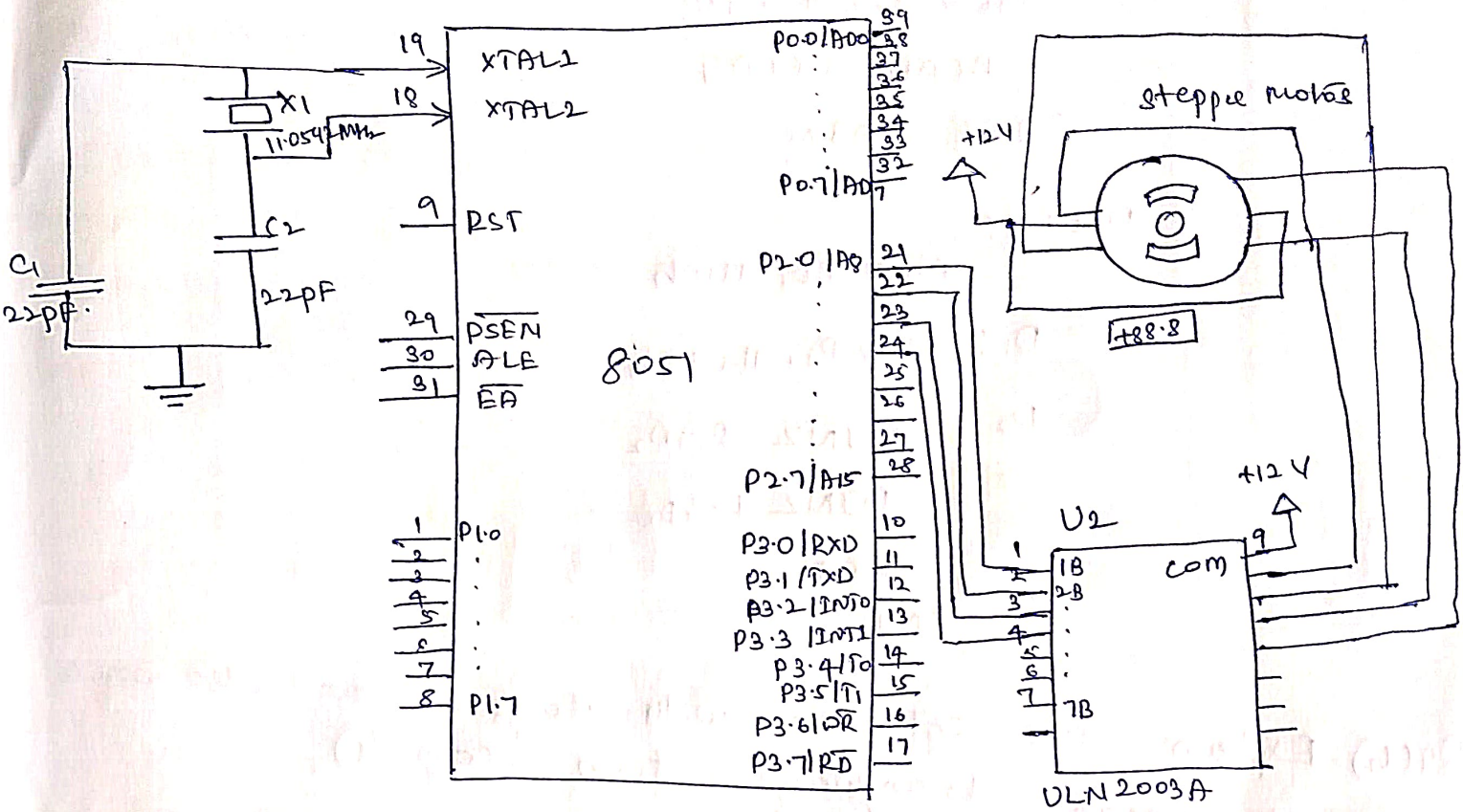
The int priority (IP) reg is used to assign high or low priority to individual interrupts.

- SFR add : B8H
- Bit addressable: Yes.
- On reset IP = 00h.

Module 5

Q9 (a) With a neat diagram write an ALP to interface stepper motor to 8051 uC. (10 marks).

Soln:



- Interfacing a steppes motor with an 8051 involves connecting the µc IIO pins (like P1.0 - P1.3) to a motor driver IC, such as the ULN2003, which when energizes the motor's coil in a specific sequence for controlled rotation controlled by an ALP. This examples uses P1.0 - P1.3 for Phase A, B, C, D resp. with a simple delay for continous rotation.

```
        Org 0000h
        Mov P1, #00h
MAIN:   Step 1: Energize Phase A. (eg P1.0)
        Mov P1, #0FEh
        ACALL Delay
        ; Step 2: Energise Phase B (P1.1)
        mov P1, #0FDh
        ACALL DELAY
        ; Step 3: Energise Phase C (P1.2)
        Mov P1, #0FBh
        ACAL DELAY
```

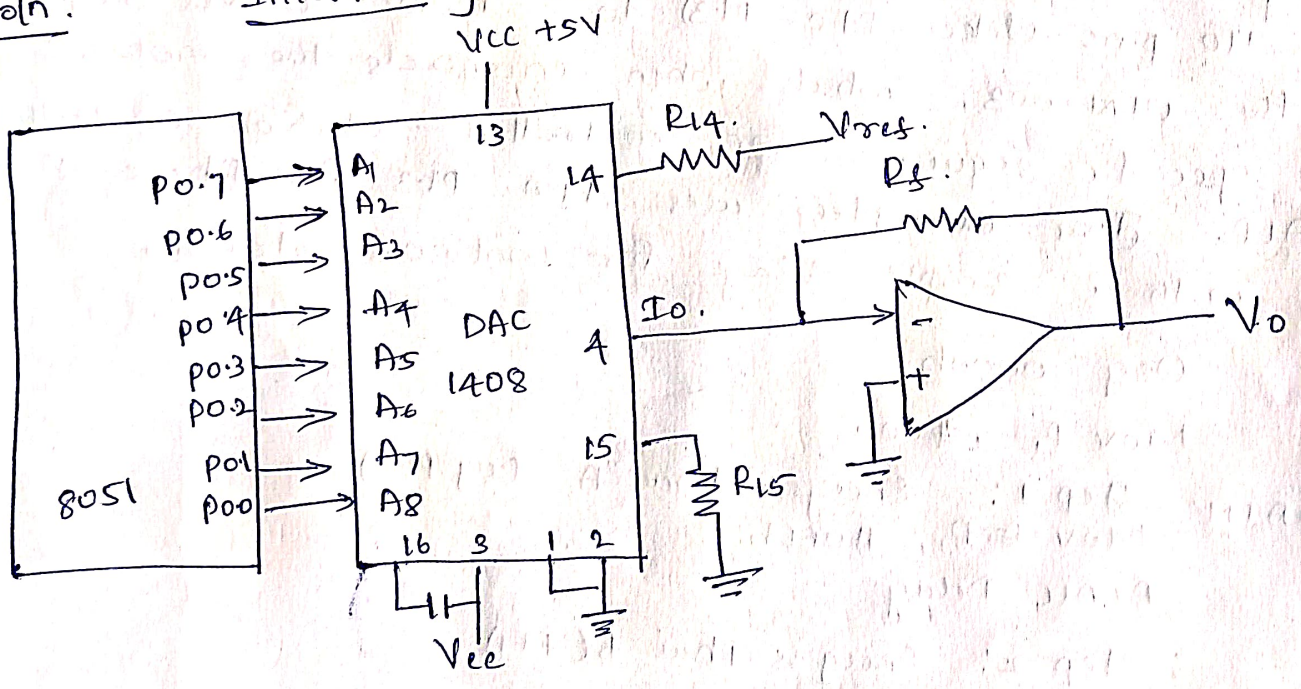
20

```
; Step 4: Energise Phase D (P1.3)
        MOV P1, #F7h;
        ACALL DELAY
        SJMP MAIN

DELAY:
        MOV R0, #10h
D1:     MOV R1, #0FFh
D2:     DJNZ R1, D2
        DJNZ R0, D1
        RET
        END.
```

Q9(b) Explain DAC interface with diagram and also write pgm to generate triangular waveform (10 marks)

Soln: Interfacing 8 bit DAC.



- DAC1408 is an 8 bit DAC.
- It produces an analog current o/p, which is converted to Vtg using an op-amp
- The digital data is provided through as I/O port of 8051

- The 8051 sends an 8 bit digital value through Port 1
- DAC 1408 converts this digital data into a proportional cln.
- The op-amp converts current into a corresponding analog vtg.
- By continously changing the digital values, different analog waveforms are generated.
- Triangular wave is obtained by digital data from 00h to FFh.

$$I_o = V_{ref}/R_{14} * (A1/2 + A2/4 + A3/8 + A4/16 + A5/32 + A6/64 + A7/128 + A8/256)$$

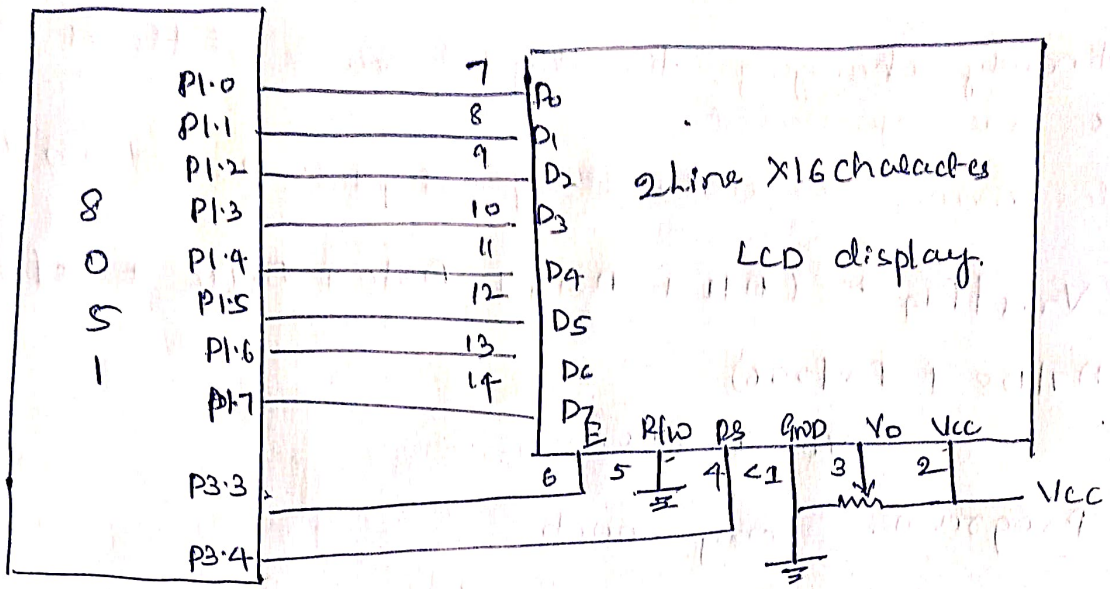
$$V_o = I_o * R_f$$

ALP Program:

```
        ORG 0000h
START:  MOV A, #00h
UP:     MOV P1, A
        ACALL DELAY
        INC A
        CJNE A, #0FFh, UP
DOWN:   MOV P1, A
        ACALL DELAY
        DEC A
        CJNE A, #00h, DOWN
        SJMP START
DELAY:  MOV R2, #0FFh
D1:     MOV R3, #0FFh
D2:     MOV R3, D2
        DJNZ R2, D1
        RET
        END
```

Q(ii) 32a) With neat diagram, write an assembly language program to interface LCD to 8051 microcontroller. (10 marks)

Soln:



Interfacing an LCD with an 8051 involves connecting control pins (RS, RW, E) and data pins (D_0-D_7 or D_4-D_7 for 4 bit mode) from LCD to an 8051 port. using specific commands to initialize & control the display and writing routines to send data/commands with appropriate delays. A circuit uses Port 1 for data pins (D_0-D_7) and Port 3 for control with R/W to ground for write operation and VEE (Pin 3) connected to variable resistor for contrast.

Circuit diagram 4bit Mode.

- Power: Vss (Pin 1) to GND, VDD (Pin 2) to +5V, Backlight (Pin 16) to +5V / GND.
- Pin 3 to middle of a Potentiometer; other ends to +5V and GND
- Control Pins: → RS : P0.0 → RW : P0.1, E → P0.2
- Data Pins D_4-D_7 to Port P2.4 to P2.7.

ALP

```
org 0000h
LJMP again
org 0030h
```

```
Main:   MOV A, #28h
        ACALL COMMAND.

        MOV A, #0C0h
        ACALL COMMAND

        MOV A, #01h
        ACALL COMMAND

        MOV A, #06h
        ACALL COMMAND

        MOV A, #80h
        ACALL COMMAND
; Display "HELLO"
        MOV A, #'H'
        ACALL DATA_DISPLAY
        MOV A, #'E'
        ACALL DATA-DISPLAY
        MOV A, #'L'
        ACALL DATA-DISPLAY
        MOV A, #'L'
        ACALL DATA-DISPLAY
        MOV A, #'O'
        ACALL DATA_DISPLAY

HERE:   SJMP HERE
        CLR P0.0
        CLR P0.1
        MOV P2, A.
        SETB P0.2
        LCALL DELAY
        CLR P0.2
        LCALL DELAY
        RET

DATA_DISPLAY
        SETB P0.0
        CLR P0.1
        MOV P2, A
        SETB P0.2
        LCALL
```

Q10) b) A door sensor is connected to the P1.1 pin and a buzzer is connected to P1.7. Write 8051 C pgm to monitor the door sensor and when it opens, sound the buzzer. The buzzer can be sound by sending a square wave of a few hundred Hz (10 marks).

24

Soln: Assumption:

- Door sensor is connected to P1.1
- Buzzer is connected to P1.7
- Door open → logic 1.
- Door CLOSED → logic 0
- Buzzer sounds when a square wave of few hundred Hz
- Polling method is used

```
#include <reg 51.h>

sbit DOOR = P1^1;
sbit BUZZER = P1^7;

void delay (void)
{
 unsigned int i;
 for(i=0; i<500; i++)
 }
    void main (void)
    {
      BUZZER = 0;
      while(1)
      { if (DOOR == 1)
      {
        BUZZER = 1;
        display ();
        BUZZER = 0;
         delay ();
```

```
    }
    else
    {
      BUZZER=0;
    }
  }
}
```

END

(CRP)

Head of the Department
Dept. of Electronic & Communication Engg
KLS V.D.I.T., HALIYAL (U.K.)